ARL-TR-8284 ● JAN 2018

US Army Research Laboratory

Semi-Automated Processing of Trajectory Simulator Output Files for Model Evaluation

by J L Cogan

NOTICES

Disclaimers

ARL-TR-8284 ● JAN 2018

US Army Research Laboratory

Semi-Automated Processing of Trajectory Simulator Output Files for Model Evaluation

by J L Cogan
Computational Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD-MM-YYYY)* January 2018	**2. REPORT TYPE** Technical Report		**3. DATES COVERED (From - To)** 1 November 2017–12 December 2017
4. TITLE AND SUBTITLE Semi-Automated Processing of Trajectory Simulator Output Files for Model Evaluation			**5a. CONTRACT NUMBER**
			5b. GRANT NUMBER
			5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) J L Cogan			**5d. PROJECT NUMBER**
			5e. TASK NUMBER
			5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory Computational Information Sciences Directorate (ATTN: RDRL-CIE) 2800 Powder Mill Road Adelphi, MD 20783-1138			**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-8284
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)			**10. SPONSOR/MONITOR'S ACRONYM(S)**
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			
13. SUPPLEMENTARY NOTES			
14. ABSTRACT Artillery trajectory simulator programs can produce detailed output that may be tailored by the user, including parameters displayed and sequence of presentation. Here we consider the General Trajectory (GTRAJ) program developed by the US Armaments Research and Development Center. The GTRAJ model has been used in the evaluation of Army meteorological systems in terms of artillery parameters such as radial miss distance and probable circular error. However, the processing of the output was tedious and is not efficient for large numbers of GTRAJ output files. The method described in this report semi-automates the process and greatly reduces the time and effort required. Results of several trial runs demonstrate the value of the method for several GTRAJ output files. The programs were written in Python (version 3.5) and would require some modification for different forms of the GTRAJ output files. However, the program described here can serve as a template for other GTRAJ output files.			
15. SUBJECT TERMS artillery meteorology, artillery trajectory simulation, python program application, artillery meteorology statistics, artillery meteorology accuracy			

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** J L Cogan
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	32	**19b. TELEPHONE NUMBER (Include area code)** 301-394-2304

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Tables

1. Introduction

Over the past several years, the US Army Research Laboratory (ARL) has supported the development and testing of Army artillery meteorological (MET) systems such as the current Profiler Virtual Module (PVM) and earlier types including the Computer, Meteorological Data–Profiler (CMD-P) and the Meteorological Measuring Set – Profiler (MMS-P). ARL supported the acquisition process by performing developmental tests of accuracy of the PVM and earlier systems. The accuracy testing compared output from the PVM (and earlier the CMD-P and MMS-P) with MET data from radiosonde observations (RAOBs) in terms of MET variables (e.g., wind speed) and measures of artillery effectiveness (e.g., radial miss distance [RMD]) as computed using a trajectory simulator program, the General Trajectory (GTRAJ) program developed by the US Armaments Research and Development Center (ARDEC). Frehlich (2006) describes a somewhat earlier version of GTRAJ, and McCoy (2012) provides some of the theory and practice behind it. The comparisons presented in a recent paper by Cogan (2017) employed similar procedures, but using spreadsheets and spreadsheet functions to produce the relevant statistics. Data from the GTRAJ output were manually entered into the spreadsheets.

This report describes software that mostly automates the analysis of the GTRAJ output and can lead to a major reduction in analysis time, perhaps to less than 20% of that needed earlier. The software consists of 3 related Python programs. The first and second programs input GTRAJ output files and either write the output from the program as a set of 2 tables in a single file for each GTRAJ file or append the output into a larger file for later processing. The first program takes an indicator from a parameter file on whether or not to append the output. It also inputs the names of the sources of MET data such as from a model or a RAOB from the same parameter file. The second program produces the same output as the first, but takes the information on whether or not to append output and the MET data source names from the command line. A third program produces overall means, medians, and standard deviations from the larger (append) file. Here the procedure for applying the software is described, samples of input and output are presented, and the code for the first and third programs are presented in Appendix A and B, respectively. The second program differs only slightly from the first and is not shown, but some differences are noted at the beginning of Appendix A.

For convenience, shorthand terms are used to identify the MET data sources for the sample output presented in this report. WRF1 refers to MET data computed from a Weather Research and Forecasting (WRF) model integration for the site of the coincident RAOB, WRF2 refers to data computed from a second WRF model integration for a site approximately 30 km to the west of the RAOB site, and R refers to data from the coincident RAOB. All 3 are contained within a single GTRAJ output file. To demonstrate the method for more than 2 MET data sources, additional GTRAJ output files labeled WRFn, where n >2, were produced by manually modifying the WRF1 or WRF2 files. Those latter files served to test the software of this report and do not represent actual atmospheric situations. The WRF model is described in Skamarock et al. (2008), and although some changes have been implemented (Lee et al. 2012; Reen et al. 2014), the basic algorithms and processes remain much the same. The sources of MET data may also include other observation systems such as for a comparison between a remote sensing system (e.g., lidar wind profiler combined with, for example, a microwave radiometer) and a RAOB.

2. Processing GTRAJ Output

The program that processes a single or multiple GTRAJ output files was developed for comparisons of results from typically a first model integration (WRF1) versus results from a coincident RAOB with results from one or more other model integrations (e.g., WRF2, WRF3) versus RAOB results. It runs on a Linux operating system (OS) computer, but should work for other OSs that can run Python 3 programs although some minor changes may be needed. The program processes a GTRAJ output text file that contains results from 2 or more simulations, where each simulation uses a computer MET message (METCM) derived from a model integration or the coincident RAOB or other source of "truth" data. US Army FM 3-09.15 (HDA 2007) describes the METCM and provides information on its application.

The output trajectory data are reported periodically along the projectile's flight path and includes the total radial distance (horizontal distance from gun to projectile), range (horizontal distance along the aiming or gun azimuth), deflection (horizontal distance perpendicular to range), height above ground (assuming smooth terrain from gun to target) of the projectile, and other user selected parameters. For these simulations the firing parameters (such as quadrant elevation, projectile weight, muzzle velocity, etc.) remained the same for all the runs. For each METCM, the simulated fires were at either 4 (i.e., toward the north, east, south, and west) or 8 (i.e., toward the north, northeast, east, southeast, etc.) directions of fire (azimuth of gun), so as to have an idea of the variation with direction (azimuth) and reduce the bias that could result from using one direction only. Consequently, each output file consisted of 3 × 4 = 12 or 3 × 8 = 24 sets of firing data output, respectively, for comparisons of trajectories computed using METCMs from 2 model integrations versus that from a RAOB. More or fewer azimuths, or more or fewer model integrations, will lead to larger or smaller sets of firing data per output file.

The meteorological data sources (e.g., WRF1, WRF2) are specified via a parameter file or as input from the command line. The former may be more convenient for processing of numerous GTRAJ output files. The operation of the program with a parameter file is straightforward in that the user only needs to type `python3` plus the program name and input filename, in that order. For the version using command-line input, after the name of the program type the name of the GTRAJ output file used as input followed by the parameter for appending or not appending the output plus the MET data source identifiers. The entire set of MET data source identifiers on the command line needs to be enclosed in either single (‘) or double (“) quotes (which forms a string in Python); either type of quotes can be used, but must be the same before and after the source names. The last source identified in the parameter file or command line list is considered the “truth” sounding, such as a RAOB. It is assumed there is only one “truth” sounding. Currently, the MET data source name must appear as the first part or prefix of the name of the METCM used for input to GTRAJ, and it must be separated from the rest of the name by an underscore (e.g., WRF1_MHX_20170514). The MET data source identifier has to relate to only one METCM within each GTRAJ output file. For example, one cannot have WRF1_MHX_2017081500 and WRF1_LMN_2017081500 within the same GTRAJ output file.

The procedure to use the program using the parameter file is shown in section 2.1 and the program code is presented in Appendix A. The command-line version of the program presented in section 2.2 primarily differs in the Python statements that are used to define the MET data sources. The Python program and the parameter file (if used) are in the same directory. The GTRAJ output files used as input may be in the same or another directory. If a separate directory is used include the file path as part of the input filename. The output file from these programs will appear in the same directory as the input file.

Each output set contains 2 tables, the first table has the RMDs for trajectories computed using METCMs from each data source (e.g., WRF1) relative to those computed using the coincident RAOB at the listed azimuths. For example, “WRF1 3200 112.4” refers to a simulation using the METCM derived from WRF1 for a firing direction or azimuth of 3200 mils (180°) and yields a RMD (WRF1 vs. R) of 112.4 m. RMD for these simulations is defined as

$$\mathrm{RMD} = ((\Delta\ \mathrm{range})^2 + (\Delta\ \mathrm{deflection})^2)^{1/2}, \qquad (1)$$

where Δ represents the difference between the value for a data source less that for the coincident RAOB (e.g., $\Delta_{2\text{-}R}$ = WRF2 value – RAOB value). The RMDs are calculated in terms of meters and then converted to percent of the radial distance computed using the METCM from the RAOB (or other "truth" sounding). Both values are printed in the first table.

The second table in the set has the means, medians, and standard deviations of the RMDs over all the azimuths for each MET data source. These quantities are calculated for the RMDs both in meters and as the percentage of radial distance; both values are printed in the second table.

2.1 Program with Parameter File

To run the program that uses a parameter file, first ensure the parameter file, named input_sources, is in the same directory as the Python program gtout.py. The parameter file has to have the parameter for appending or not appending the output (a = append) followed by the identifiers of the MET data sources space-delimited and listed on the first line. Normally, the MET data sources are listed in the order they appear in the GTRAJ output file. For example, the line in the parameter file may list 'n WRF1 WRF2 R' for not to append (i.e., a single output file), METCMs from 2 model integrations and the coincident RAOB. They must have the same names as the prefixes in the METCM files named in the GTRAJ output file (e.g., WRF1). The "truth" sounding is the last MET data source (e.g., in the previous example R indicating RAOB is the "truth" sounding). Section 2.3 has additional information on the selection, number, and ordering of the MET data source identifiers.

On the command line, enter the name of the program and the input GTRAJ file.

```
python3  gtout.py INPUT_FILE ,
```

where INPUT_FILE is the name of the GTRAJ output file (input to program). The file produced by the program has the addition of _out to the input file name for the case of not appending and has the name output-tables for appending.

For example,

```
python3  gtout.py DEN_2017-09-17-00.out ,
```

produces an output file DEN_2017-09-17-00.out_out (not appending) or output_tables (appending). The filenames for GTRAJ output are chosen by the user and do not have to have the .out extension. However, the default name is gtraj.out, and that extension is often used. The name for the aforementioned examples could have been DEN_2017-09-17-00, DEN_2017-09-17_test, or some other name.

2.2 Program with Source Names on Command Line

The procedure for the version with entry of the MET data source names via the command line closely follows that for the one using the parameter file ("cl" indicates the command line version). The program reads the command-line list that contains the append parameter and then the source names as a single string; the source names must be enclosed in single or double quotes.

On the command line, enter the following:

```
python3 gtoutcl.py INPUT_FILE X 'SOURCE-1 SOURCE-2 … SOURCE-N' ,
```

where INPUT_FILE is the name of the GTRAJ output file as before, X is the append parameter (a = append and any other character [e.g., n] = do not append), and SOURCE-1 … SOURCE-N are generic names for the sources of the MET data in the GTRAJ output. SOURCE-N is the name of the source of the "truth" sounding (e.g., RAOB). The output has the addition of _out to the input file name (not appending) or the name output_files (appending) as before.

For example,

```
python3 gtoutcl.py DEN_2017-09-09-17-00.out n 'WRF1 WRF2 R',
```

where n indicates single file output (a = append), and WRF1, WRF2, and R indicate METCMs that were produced from the 2 model integrations and from RAOB data. In this example, the program will produce an output file with the same name as for the gtout.py program without appending (DEN_2017-09-17-00.out_out). Note that one has to enclose the source names in either double or single quotes.

2.3 A Word on METCM Filenames

The filenames for the METCMs used by the GTRAJ program from whatever source must begin with the source identifier (e.g., WRF2) followed by an underscore (_) and the rest of the name, usually with the site identifier followed by a string indicating the date and time (e.g., WRF1_LMN_2017081500 for WRF integration 1 for Lamont, OK, at 00 Coordinated Universal Time [UTC] on 15 August 2017). The last source name in the parameter file or command-line list is considered the "truth" sounding. However, if a different order of the prefixes in the naming convention is chosen (e.g., LMN_WRF1_2017081500), minor changes to only a few statements should enable the programs to run properly.

If one or more of the MET data source names in the parameter file (or on the command line) is/are not the same as in the GTRAJ output file the program will partly run and then produce an error message (KeyError: followed by a string, which may represent a number). That includes the situation where more sources appear in the parameter file or command line than are in the GTRAJ output file (i.e., produces a KeyError error message). If the order that the MET data source names appear is different, the program will run, but the order in which the output data are listed in the output tables will be different. For example, WRF1, R, WRF2 as sources in that order will lead to comparisons of WRF1 to WRF2 and R to WRF2, treating data from WRF2 as the "truth" values. If the parameter file or the command line has fewer data sources than the GTRAJ file, then the program will run normally and produce output for the sources listed in the parameter file or command line. However, at least 2 MET data sources, one of which is considered the "truth" source, are required since differences are computed.

The ability of the program to produce output whatever the order of the MET data source names allows additional comparisons to be made. For example, one may want to compare results from one type of model output (e.g., WRF) with another type of model output (e.g., GFS). Also, one may want to investigate the changes in RMD when one of the other sources is used as the "truth" sounding. Another potential comparison could involve use of different azimuths or number of azimuths. For example, a comparison of the use of 4 azimuths versus 8 or 16 azimuths on mean or median values for a given site. Another may consider the effect of one set of directions versus another (e.g., north and west vs. south and east).

3. Difference Statistics from Multiple Tables

A third program, gtstats.py, is used to compute basic differences statistics for the RMDs from all tables of the individual RMDs in the output_tables file. Means, medians, and standard deviations of the RMDs for each MET data source over all azimuths from all tables of individual RMDs for each site are computed from the respective values. An individual RMD refers to an RMD derived from a single data source relative to the coincident RAOB (or other "truth" sounding) calculated for one site at one time at one azimuth (e.g., an RMD for WRF1 vs. RAOB computed for MHX [Newport, VA] at 2017082300 for an azimuth of 1600 mils). The MET data sources are defined from the identifiers in the first line of output_tables. The procedure for running this program, gtstats.py is as follows:

python3 gtstats.py INPUT_FILE,

where INPUT_FILE is the output file from gtout.py or gtoutcl.py, that is, output_tables if using the default name.

For example,

python3 gtstats.py output_tables,

where the names of the MET data sources are extracted from the first line of output_tables.

The output filename is RMD_statistics_out, which is the default. That may be changed in the program by modifying one statement.

If the MET data sources in output_tables are not the same in the tables from consecutive program runs after the first one, then the program will ignore those not listed in the first line of the first run. Therefore, it is important to ensure the first line of the first run (top header line of the first set of 2 tables) is correct. If the azimuths are not the same, then the program will run, but the results may not be valid. The program will run for one or more azimuths for each data source, but there must be at least 3 samples (individual RMDs) for computation of standard deviation. When there are fewer than 3 samples for a data source, the program will print a message to the screen before ending normally (NO STANDARD DEVIATION COMPUTED FOR DATA LIST N - LESS THAN 3 SAMPLES !!, where N is the number of the data source [e.g., 0 or 1 = WRF1 or WRF2, respectively]). The values for standard deviation will be listed as -999, the missing data indicator. Appendix B contains details on this program.

4. Input and Output Samples

The type of input file from GTRAJ is the same for gtout.py and gtoutcl.py (i.e., the same input file may be used for both programs). Table 1 presents the first section of a sample GTRAJ output file. Only the listing for 1 of 8 azimuths for 1 of 3 MET data sources are shown here (24 listings altogether) given the size of the file and that subsequent sets have the same format and parameters. The output contains trajectory data, intermediate values (e.g., wind velocity in terms of the components in the range and deflection directions), values of many firing and other parameters most of which were not changed from their default values, and so on. The output shown here has trajectory values (e.g., range and deflection) every 10 s. The user can change the frequency of trajectory output and add or remove output of some variables (e.g., speed of the projectile in terms of Mach number) when executing the GTRAJ program.

Table 1 Output from the first of 24 listings within a single GTRAJ output file. The GTRAJ output file included 3 MET data sources over 8 azimuths. Note that in GTRAJ the radial distance (distance from gun to target) is called range. In the output listing at the bottom, the actual range is named E1, deflection E3, and height above the Earth's surface E2.

```
AERODYNAMICS: C:\GTRAJ3v97\DB\aero\155.aero  M109A/M198/M777A2 M795                                          GTRAJ V9.7
 BALLISTICS:   C:\GTRAJ3v97\DB\bal\m109a5_a6_a7.bal  M109A7   M795        HE                                  20-NOV-17
 FCI:          FCI 155-AR-A                 4 Jan 14                                                          15:47:00
 HOWITZER/CANNON MODEL: M109A7/M284
 FUZE:
 FUZE MODE:

   STD WT    ACT WT    INIT WT    TWIST    DIAMETER  C.G. REF    C.G.   AXIAL MOM   WT SQ   TUBE LEN  DIA BASE   FBT-N    FBT-P
    LB        LB        LB       CAL/REV     MM       FEET       FEET    LB*FT^2     LB      FEET
   103.5     103.5      0.0        20.0     155.0      0.0        0.0     4.0058     1.1      0.0       0.0       0.0      0.0

   TIME STP   TS MIN   TS MAX  TIME STPB   TS MINB  TS MAXB   K-FACTOR
     %         SEC      SEC       %          SEC      SEC
    3.0       0.01      1.0      0.75      0.0025     0.25   -0.166667

    EARTH MOD   CORIOLIS    GRAVITY      WINDAGE JUMP   CORR MV FOR PWT     TRAJ MODEL

     CURVED        YES        STD              YES             NO                MPMS

    ELEV      THETA  BAL COEFF  LIFT FAC     MFF       FPDM    Q FACTOR    HOB       CSM      DELTA U    KD SM
    MILS      MILS    LB/IN^2                                              M      LB/IN^2      M/S
    800.0      0.0    2.934668  0.97176      1.0        0.0       1.2      0.0       0.0        0.0       0.0

  CHARGE       U0        AZ        T0        X0        Y0        Z0       LAT       NO     PROP TEMP  EFC COUNT    AOS
              M/S       MILS       SEC        M         M         M       DEG     REV/SEC    DEG F                 MILS
   5H        792.0      0.0       0.0       0.0      328.0      0.0      52.81      0.0       70.0       0.0        0.0

   PRT ON    PRT INT  STP CODE   STOP ON   FINAL T  FINAL TIME  FINAL FS  FINAL E1  FINAL E2  FINAL E3  FINAL RG  FNL E2E1 ELEV MAX
              SEC                            SEC       SEC        SEC        M         M         M         M        M       MILS
    TIME      10.0       4         N/A       60.0      60.0       60.0    10000.0    328.0      0.0     10000.0     0.0

   BAL AD    BAL AT    BAL PRESS  B RG WD   B VERT WD   B CR WD  HGT MDP  GRID DEC
     %         %          %       M/SEC      M/SEC      M/SEC      M       MILS
   100.0     100.0      100.0      0.0        0.0        0.0     307.55     0.0

 MET FILE: C:\GTRAJ3v97\met\WRF_test\WRF1_LMN_2017081500_gt

      HGT    AIR TEMP   AIR DEN    AIR PRESS   HGT     RANGE WD  CROSS WD
       M       DEG K    KG/M**3       MB        M        M/S       M/S
      307.6 307.20000   1.10226      972.0   307.550     3.306    -2.452
      407.6 305.90000   1.09555      962.0   407.550     3.798    -2.647
      657.5 302.90000   1.07535      935.0   657.550     4.386    -2.688
     1057.5 298.80000   1.04114      893.0  1057.550     4.328    -1.645
     1557.5 294.30000   0.99787      843.0  1557.550     2.523     0.502
     2057.6 290.90000   0.95205      795.0  2057.550    -0.382     2.022
     2557.6 288.80000   0.90469      750.0  2557.550    -2.663     1.561
     3057.6 285.50000   0.86268      707.0  3057.550    -4.036     0.803
     3557.6 281.50000   0.82296      665.0  3557.550    -4.082     0.524
     4057.6 278.00000   0.78445      626.0  4057.550    -5.114     0.554
     4557.6 275.10000   0.74460      588.0  4557.550    -5.620     0.665
     5057.6 272.40000   0.70722      553.0  5057.550    -5.550     1.104
     5807.6 267.50000   0.65506      503.0  5807.550    -4.041     3.962
     6807.6 261.00000   0.58996      442.0  6807.550    -1.057     7.644
     7807.6 253.60000   0.53162      387.0  7807.550     1.692    12.230
     8807.5 245.80000   0.47762      337.0  8807.550     3.528    16.080
     9807.5 238.20000   0.42851      293.0  9807.550     4.564    19.009
    10807.5 231.60000   0.38056      253.0 10807.550     7.008    25.817
    11807.5 225.80000   0.33633      218.0 11807.550     8.491    31.278
    12807.5 219.90000   0.29625      187.0 12807.550     8.573    32.853
    13807.5 213.60000   0.26095      160.0 13807.550     7.256    31.587
    14807.5 209.40000   0.22626      136.0 14807.550     4.273    25.365
    15807.5 206.90000   0.19363      115.0 15807.550     0.323    16.459
    16807.5 205.60000   0.16605       98.0 16807.550    -1.339     9.682
    17807.5 207.80000   0.13915       83.0 17807.550     1.752     6.454
    18807.5 210.40000   0.11590       70.0 18807.550     4.155     3.841
    19807.5 212.40000   0.09841       60.0 19807.550     2.376     0.984
    21307.5 213.61211   0.07740       47.5 21307.550     2.376     0.984
    23307.5 215.55918   0.05625       34.8 23307.550     2.376     0.984
    25307.5 217.50504   0.04101       25.6 25307.550     2.376     0.984
    27307.5 219.44968   0.02999       18.9 27307.550     2.376     0.984
    29307.5 221.39310   0.02200       14.0 29307.550     2.376     0.984

 * END OF HEADER *
     TIME       RANGE       E1         E3         E2       MACH NO
      SEC       METERS     METERS     METERS     METERS

     0.00000      0.000      0.000      0.000    328.000    2.24822
    10.00000   4661.440   4661.428     10.361   4550.490    1.53762
    20.00000   8237.031   8236.896     47.322   6912.811    1.13943
    30.00000  11295.184  11294.535    121.049   8009.237    0.93373
    40.00000  14125.683  14123.716    235.700   8081.120    0.87811
    50.00000  16808.259  16803.729    390.237   7220.901    0.91591
    60.00000  19304.585  19296.092    572.598   5506.948    0.98709
    70.00000  21507.586  21494.053    762.855   3101.180    1.01736
    79.67057  23288.699  23269.753    939.208    328.000    1.00046
   * END OF Data *
```

The output files from gtout.py or gtoutcl.py are the same. Table 2 shows the output tables for the GTRAJ file used for Table 1. The upper table has individual RMDs in terms of meters and percent radial distance (named range in GTRAJ) for each

MET data source or system and azimuth (mils). The lower table has the mean and median values for each MET data source in terms of meters and percent radial distance. Although percent range is the common terminology, in these simulations it is actually percent radial distance. The RMD in m is divided by the radial distance computed for the RAOB to obtain the percent radial distance.

Table 2 RMDs (data source – RAOB) computed for Lamont, Oklahoma, on 20170815 at 00 UTC for each MET data source and azimuth (upper table), and mean and median values for each MET data source (lower table). Azimuth is in mils and % refers to percent radial distance (aka % range).

```
RMDs for WRF1 and WRF2 less R
 for site and date/time: LMN_2017081500

System    Azimuth     RMD (m)      RMD (%)
WRF1         0.0       44.7        0.192
WRF2         0.0       26.6        0.114
WRF1       800.0       48.9        0.207
WRF2       800.0       38.0        0.161
WRF1      1600.0       34.7        0.146
WRF2      1600.0       35.2        0.148
WRF1      2400.0       10.4        0.044
WRF2      2400.0       17.9        0.076
WRF1      3200.0       19.3        0.083
WRF2      3200.0        6.7        0.029
WRF1      4000.0       25.4        0.111
WRF2      4000.0       19.9        0.087
WRF1      4800.0       10.6        0.047
WRF2      4800.0       14.8        0.065
WRF1      5400.0       14.0        0.061
WRF2      5400.0        1.3        0.006

System         Mean RMD (m)  Mean RMD (%)
WRF1_mean_RMD       26.02       0.111
WRF2_mean_RMD       20.06       0.086
WRF1_median_RMD     22.39       0.097
WRF2_median_RMD     18.90       0.081
```

Table 3 shows the output for 2 instances of gtout.py (or gtoutcl.py) where the set of 2 tables from each program execution are appended. There is no preset limit to the number of table sets that can be appended. However, they should relate to the same MET data sources and have the same azimuths.

Table 3 Output from gtout.py for 2 sites, 2 sources of MET data, and 8 azimuths

```
RMDs for WRF1 and WRF2 less R
 for site and date/time: ETGB_2016-02-07-00

System    Azimuth     RMD (m)     RMD (%)
WRF1         0.0      149.0       0.638
WRF2         0.0      198.3       0.849
WRF1       800.0      167.6       0.715
WRF2       800.0      122.5       0.523
WRF1      1600.0      267.5       1.170
WRF2      1600.0      234.7       1.027
WRF1      2400.0      262.9       1.195
WRF2      2400.0      274.5       1.248
WRF1      3200.0      166.1       0.779
WRF2      3200.0      195.4       0.916
WRF1      4000.0      144.1       0.677
WRF2      4000.0      122.5       0.576
WRF1      4800.0      230.9       1.056
WRF2      4800.0      218.9       1.002
WRF1      5400.0      248.5       1.104
WRF2      5400.0      272.3       1.210

System         Mean RMD (m)  Mean RMD (%)
WRF1_mean_RMD      204.57      0.917
WRF2_mean_RMD      204.89      0.919
WRF1_median_RMD    199.24      0.918
WRF2_median_RMD    208.62      0.959


RMDs for WRF1 and WRF2 less R
 for site and date/time: LMN_2017081500

System    Azimuth     RMD (m)     RMD (%)
WRF1         0.0       44.7       0.192
WRF2         0.0       26.6       0.114
WRF1       800.0       48.9       0.207
WRF2       800.0       38.0       0.161
WRF1      1600.0       34.7       0.146
WRF2      1600.0       35.2       0.148
WRF1      2400.0       10.4       0.044
WRF2      2400.0       17.9       0.076
WRF1      3200.0       19.3       0.083
WRF2      3200.0        6.7       0.029
WRF1      4000.0       25.4       0.111
WRF2      4000.0       19.9       0.087
WRF1      4800.0       10.6       0.047
WRF2      4800.0       14.8       0.065
WRF1      5400.0       14.0       0.061
WRF2      5400.0        1.3       0.006

System         Mean RMD (m)  Mean RMD (%)
WRF1_mean_RMD       26.02      0.111
WRF2_mean_RMD       20.06      0.086
WRF1_median_RMD     22.39      0.097
WRF2_median_RMD     18.90      0.081
```

The output from the program, gtstats.py, that computes overall means, medians, and standard deviations over all the RMDs for each source of MET data (e.g., WRF1) obtains input from the file output_tables (if the default name is used). As noted previously, the names of the MET data sources are extracted from the first line of the input file. Table 4 presents a sample of output from the gtstats.py program where the input file (output_tables) has the 2 sites (ETGB and LMN) and 8 azimuths of Table 3.

Table 4 Output from gtstats.py, where the input file (output_tables) contained tables for the 2 sites and 8 azimuths of Table 3

```
Means, medians, and standard deviations of 16 individual RMDs

        Means
 Data Source      (m)    (%Range)
    WRF1       115.29   0.514
    WRF2       112.47   0.502

        Medians
 Data Source      (m)    (%Range)
    WRF1        96.50   0.422
    WRF2        80.25   0.342

        Std Deviations
 Data Source      (m)    (%Range)
    WRF1        99.58   0.448
    WRF2       103.92   0.468
```

5. Summary and Conclusion

This report presented short descriptions of 2 Python 3 programs that extract trajectory information and certain parameters such as site and azimuth from GTRAJ output files and create tables of RMD by source of MET data and azimuth as well as statistics for each source over all azimuths. The first obtains the names of the sources of MET data from a parameter file, and the second takes the names of the MET data sources from the command line. By setting one parameter in the parameter file or the command line, each program can append the output from processing 2 or more GTRAJ output files to a single file rather than one file for each GTRAJ output file processed. For many applications, use of the parameter file version could help reduce time and effort, and reduce the opportunity for incorrect entries (e.g., typos on the command line). The third program described in this report computes the mean, median, and standard deviation for each source of MET data over all the azimuths and sites.

An important consideration is the greatly decreased time to process the GTRAJ output file data. Based on some sample runs, less than 20% of the time is needed compared with using spreadsheets for the calculation of the statistics. However, care must be taken to avoid processing a GTRAJ output file more than once, thereby creating duplicate tables in the file containing the appended data. Another consideration concerns verifying that the azimuths are the same for all tables in the file with the appended data. If not the same, the program for the overall statistics will run, but the results could be misleading.

Currently the processing assumes the GTRAJ file contains values for one or more sources of MET data that will be compared with values from a “truth” source. The sources of MET data (as for this report) may be numerical weather prediction models at any scale, global to microscale. However, the sources also may include observation systems. The “truth” source of MET data usually is a coincident RAOB, but also could be another source of a sounding (e.g., radar or lidar wind profiler combined with a microwave radiometer). The programs described herein provide a means to process GTRAJ output files far more rapidly while reducing the chance of some types of data entry errors. However, GTRAJ output having, for example, a different ordering of the trajectory output columns or other related changes would require modifications to the programs that should not require extensive effort. Nevertheless, the programs of this report can serve as templates for processing other variants of the GTRAJ output files or output files from other trajectory simulator programs.

6. References

Cogan J. Model evaluation using ballistic trajectories and preliminary mesoscale model accuracies with age of global model initialization data. Meteorol Appl. 2017. doi:10.1002/met.1684.

Frehlich R, Sharman R, Clough C, Padovani M, Fling K, Boughers W, Walton, WS. Effects of Atmospheric Turbulence on Ballistic Testing. J Appl Meteo Climato. 2006;47:1539–1549.

Lee JA, Kolczynski WC, McCandless TC, Haupt SE. An objective methodology for configuring and down-selecting an NWP ensemble for low-level wind prediction. Mon Wea Rev. 2012;140:2270–2286.

McCoy RL. Modern exterior ballistics: the launch and flight dynamics of symmetric projectiles. Atglen (PA): Schiffer Publishing Ltd.; 2012. p. 328. ISBN: 978-0-7643-3825-0. www.schifferbooks.com.

Reen BP, Stauffer DR, Davis KJ. Land-surface heterogeneity effects in the planetary boundary layer. Boundary-Layer Meteorology. 2014;150:1–31.

Skamarock WC, Klemp JB, Dudhia J, Gill DO, Barke DM, Duda MG, Huang XY, Wang W, Powers JG. A description of the advanced research WRF version 3. Boulder (CO): National Center for Atmospheric Sciences (US); 2008. Report No.: NCAR/TN-475+STR.

[HDA] Headquarters, Department of the Army. Artillery meteorology. Washington (DC) Headquarters, Department of the Army; 2007. Field Manual No.: FM 3-09.15. p. 270.

INTENTIONALLY LEFT BLANK.

Appendix A. Python 3 Code (gtout.py)

This Appendix has the Python 3 code (gtout.py) for extraction of relevant General Trajectory (GTRAJ) simulation values from a GTRAJ output file and computation of mean, median, and standard deviation values for 2 or more sources of computer meteorological message (METCM) data over a set of several azimuths. The parameter on whether or not to append the output (i.e., a = append) and the data sources (e.g., WRF1) are read from a parameter file with the append parameter as the first item in a line that includes the METCM data sources. For the command-line version (gtoutcl.py), the append parameter follows the input filename, which in turn is followed by a single string that contains all the sources of METCM data.

```
#!/bin/env python3

import re
import sys
from collections import defaultdict
import string
import statistics
import os
import ntpath

#NOTE: sys.argv[0] is the program (e.g., gtout.py).
with open('input_sources', "r") as p:
    data_sources = p.readline() #Reads one line that has the 3 data sources (e.g., GFS).
    source_list = data_sources.split()
    append_or_not = source_list[0] #Read first item in list: the append or not indicator (a for
append).
    del source_list[0]
    source_len = len(source_list)

with open(sys.argv[1], "r") as f:
    input_data = f.readlines()  #Everything read in.

if append_or_not == 'a':
  output_file = 'output_tables'
else:
  output_file = sys.argv[1] + "_out"
print('\nReading from file: ', sys.argv[1], "\n\n")

#Prepare variable names.

dataval_list = []    # Set up empty list for data values.
difval_list = []       # Set up empty list for difference values.
RMDvalue = []       # Empty lists for RMD.
RMDpctvalue = [] # Empty list for RMD %.

for n in range(0, source_len):
  dataval_list.append('data_val_v'+str(n))
  dataval_list[n] = defaultdict(dict)
  difval_list.append('dif_v'+str(n+1)+'-v'+str(source_len))
```

```
  difval_list[n] = defaultdict(dict)
  difval_list.append('dif_v'+str(n+1)+'-v'+str(source_len))
  difval_list[n] = defaultdict(dict)
  RMDvalue.append('RMD_v'+str(n+1)+'-v'+str(source_len))
  RMDvalue[n] = defaultdict(dict)
  RMDpctvalue.append('RMDpct_v'+str(n+1)+'-v'+str(source_len))
  RMDpctvalue[n] = defaultdict(dict)

# Set up a set and an empty list for azimuths.
az_vals = set()
az_values = []

#Define the previous line (any string should work).
prevline = 'first next line'

for currentline in input_data:

# Find the azimuth before reading data lines.
   match = re.search('DEG F', prevline)
   if match:
     data_list = currentline.split()
     azimuthval = data_list[2]
     azimuth = float(azimuthval)
     azint = int(azimuth)
     azstr = str(int(azimuth))
     az_vals.add(azimuth)
     az_values.append(azimuth)

# Find site and type of data (e.g., range), and read in data.

   match = re.search('MET FILE:', currentline)
   if match:
     metfile = ntpath.basename(currentline)
     sys_list = re.split("_", metfile)
     sys = sys_list[0]
     site = sys_list[1]
     site_datetime = sys_list[1] + "_" + sys_list[2]
     for k in range(0, source_len):
       if sys == source_list[k]:     # k is set to the source's index number in source_list.
         break
     site_and_az = site + "_" + azstr

   match = re.search('END OF Data', currentline)
   if match:
     data_list = prevline.split()
     dataval_list[k]['radial_dist'][str(int(azimuth))] = data_list[1]
     dataval_list[k]['range'][str(int(azimuth))] = data_list[2]
     dataval_list[k]['deflection'][str(int(azimuth))] = data_list[3]
   else:
     prevline = currentline

sorted_azvals = sorted(az_vals)     # Sort on azimuths.
#
```

```
# Compute difference values.
#
# Create empty lists for the various statistics for RMD in m and % of radial distance.
sum_RMD_list = []
sum_RMDpct_list = []
mean_RMD_list = []
mean_RMDpct_list = []
median_RMD_list = []
median_RMDpct_list = []

for n in range(0, source_len-1):
  RMD_list = []
  RMDpct_list = []
  for azim in sorted_azvals:
    radial_dist_dif = float(dataval_list[n]['radial_dist'][str(int(azim))]) -
float(dataval_list[source_len-1]['radial_dist'][str(int(azim))])
    range_dif = float(dataval_list[n]['range'][str(int(azim))]) - float(dataval_list[source_len-
1]['range'][str(int(azim))])
    deflection_dif = float(dataval_list[n]['deflection'][str(int(azim))]) -
float(dataval_list[source_len-1]['deflection'][str(int(azim))])
    RMDvalue[n][str(int(azim))] = (range_dif*range_dif + deflection_dif*deflection_dif)**0.5
    RMDpctvalue[n][str(int(azim))] = float(RMDvalue[n][str(int(azim))]) /
float(dataval_list[source_len-1]['radial_dist'][str(int(azim))]) * 100
    RMD_list.append(RMDvalue[n][str(int(azim))])
    RMDpct_list.append(RMDpctvalue[n][str(int(azim))])

#  Mean and median values for all azimuths for each input source.
  mean_RMD = statistics.mean(RMD_list)
  mean_RMDpct = statistics.mean(RMDpct_list)
  median_RMD = statistics.median(RMD_list)
  median_RMDpct = statistics.median(RMDpct_list)
  mean_RMD_list.append(mean_RMD)
  median_RMD_list.append(median_RMD)
  mean_RMDpct_list.append(mean_RMDpct)
  median_RMDpct_list.append(median_RMDpct)
#
#OUTPUT SECTION: output generated here although some output strings composed earlier in
program.
#
if append_or_not == 'a':
  x = 'a'
else:
  x = 'w'
with open(output_file, x) as fo:
  print('Writing to file: ', output_file, "\n\n")
  title_string = 'RMDs for '
  for n in range(0, source_len-1):
   if(n <  source_len-2):
    title_string = title_string + source_list[n] + ' and '
   else:
    title_string = title_string + source_list[n]
  title_string =  title_string + ' less ' + source_list[source_len-1]
```

```
  header_string='{0:42s}\n{1:19s}{2:5s}\n'.format(title_string,' for site and date/time: ',
site_datetime)
  fo.write(header_string)
  header_string='\n{0:43s}\n'.format('System    Azimuth     RMD (m)     RMD (%)')
  fo.write(header_string)

  for azim in sorted_azvals:
    for n in range(0, source_len-1):
      rmd_string = '{0:8s} {1:7.1f}   {2:8.1f}    {3:8.3f}\n'.format(source_list[n], float(azim),
float(RMDvalue[n][str(int(azim))]), float(RMDpctvalue[n][str(int(azim))]))
      fo.write(rmd_string)

  header_string='\n{0:43s}\n'.format('System        Mean RMD (m)  Mean RMD (%)')
  fo.write(header_string)

  for n in range(0, source_len-1):
    mean_string = '{0:16s} {1:8.2f}   {2:8.3f}\n'.format((source_list[n]+'_mean_RMD'),
float(mean_RMD_list[n]), float(mean_RMDpct_list[n]))
    fo.write(mean_string)
  for n in range(0, source_len-1):
    median_string = '{0:16s} {1:8.2f}   {2:8.3f}\n'.format((source_list[n]+'_median_RMD'),
float(median_RMD_list[n]), float(median_RMDpct_list[n]))
    fo.write(median_string)
  if x == 'a':
    fo.write('\n\n')
```

INTENTIONALLY LEFT BLANK.

Appendix B. Python 3 Code (gtstats.py)

This appendix has the Python 3 code (gtstats.py) for computation of overall means, medians, and standard deviations of all radial miss distances (RMDs) for each source of meteorological (MET) data. The identifiers of the MET data sources (e.g., WRF1) are read from the first line of the input file (e.g., output_tables).

```
#!/bin/env python3

import re
import sys
from collections import defaultdict
import string
import statistics

#NOTE: sys.argv[0] is the program (e.g., gtstats.py).
with open(sys.argv[1], "r") as g:
    first_line = g.readline()       #Only read first line.
    input_data = g.readlines()  #All other lines read in.

match = re.search('RMDs for', first_line)
if match:
    first_list = first_line.split()
    length_firstlist = len(first_list)

output_file = 'RMD_statistics_out'
print('\nReading from file: ', sys.argv[1], "\n")

source_list = []
for n in range(2, length_firstlist, 2):
  source_list.append(first_list[n])

variable_val = defaultdict(dict)
variable_pctval = defaultdict(dict)
site_str = defaultdict(dict)

var_list = []
length_list = []
length_pctlist = []
var_mean_list = []
varpct_mean_list = []
var_median_list = []
varpct_median_list = []
var_stdev_list = []
varpct_stdev_list = []
#
# Fill lists with values from input file and compute various statistics.
#
prevline = "prev line"

for n in range(0, len(source_list)-1):
 variable_datalist = []
 variable_pctlist = []
```

```
 for currentline in input_data:
   match = re.search('Azimuth', prevline)
   site_str[n] = source_list[n] + '   '
   #print('site_str[n] ', site_str[n])
   match = re.search(site_str[n], currentline)
   if match:
       var_list = currentline.split()
       varlist_len = len(var_list)
       variable_val[site_str[n]] = float(var_list[2])
       variable_pctval[site_str[n]] = float(var_list[3])
       variable_datalist.append(variable_val[site_str[n]])
       length_list.append(len(variable_datalist))
       variable_pctlist.append(variable_pctval[site_str[n]])
       length_pctlist.append(len(variable_pctlist))
 #Compute various statistics.
 var_mean = statistics.mean(variable_datalist)
 varpct_mean = statistics.mean(variable_pctlist)
 var_median = statistics.median(variable_datalist)
 varpct_median = statistics.median(variable_pctlist)
 if(len(variable_datalist) > 2):
   var_stdev = statistics.stdev(variable_datalist)
   varpct_stdev = statistics.stdev(variable_pctlist)
 else:
   var_stdev = -999
   varpct_stdev = -999
   no_stdev_string = 'NO STANDARD DEVIATION COMPUTED FOR DATA LIST ' + str(n) + ' - LESS
THAN 3 SAMPLES !!'
   print(no_stdev_string) # Since n starts at 0 it's the first data list.

 #Append statistical values to respective lists.
 var_mean_list.append(var_mean)
 varpct_mean_list.append(varpct_mean)
 var_median_list.append(var_median)
 varpct_median_list.append(varpct_median)
 var_stdev_list.append(var_stdev)
 varpct_stdev_list.append(varpct_stdev)

 prevline=currentline

#Check for equal number of items in data lists.
for n in range(0, len(source_list)-1):
  if length_list[n] != length_pctlist[n]:
   print("List length mismatch!\n")
   print("variable_datalist = ", len(length_list), " variable_pctlist = ", len(length_pctlist))
#
# Output section: mean and median values.
#
with open(output_file, "w") as fo:
    print('Writing to file: ', output_file, "\n")
    header_string = 'Means, medians, and standard deviations of ' + str(len(variable_datalist)) + '
individual RMDs\n'
    fo.write(header_string)
```

```
    header_string = '\n       Means\n Data Source     (m)   (%Range)\n'
    fo.write(header_string)
    for k in range(0, len(source_list)-1):
      data_string = '    {0:9s} {1:7.2f} {2:7.3f}\n'.format(source_list[k], float(var_mean_list[k]),
float(varpct_mean_list[k]))
      fo.write(data_string)
    header_string = '\n       Medians\n Data Source     (m)   (%Range)\n'
    fo.write(header_string)
    for k in range(0, len(source_list)-1):
      data_string = '    {0:9s} {1:7.2f} {2:7.3f}\n'.format(source_list[k], float(var_median_list[k]),
float(varpct_median_list[k]))
      fo.write(data_string)
    header_string = '\n       Std Deviations\n Data Source     (m)   (%Range)\n'
    fo.write(header_string)
    for k in range(0, len(source_list)-1):
      data_string = '    {0:9s} {1:7.2f} {2:7.3f}\n'.format(source_list[k], float(var_stdev_list[k]),
float(varpct_stdev_list[k]))
      fo.write(data_string)
```

List of Symbols, Abbreviations, and Acronyms

ARL	US Army Research Laboratory
ARDEC	US Armaments Research and Development Center
CMD-P	Computer, Meteorological Data–Profiler
GTRAJ	General Trajectory
MET	meteorological
METCM	computer MET message
MMS-P	Meteorological Measuring Set – Profiler
PVM	Profiler Virtual Module
RAOB	radiosonde observation
RMD	radial miss distance
WRF	Weather Research and Forecasting

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIR ARL IMAL HRA RECORDS MGMT RDRL DCL TECH LIB
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	ARL RDRL CIE J L COGAN